GPFS Overview – Part II

March 22, 2002

Bill Loewe <wel@llnl.gov>
Mark Grondona <grondona@llnl.gov>

This work was performed under the auspices of the
U.S. Department of Energy
by the University of California,
Lawrence Livermore National Laboratory
under contract No. W-7405-Eng-48.

Topics

PART I:

- General GPFS Architecture and Functionality

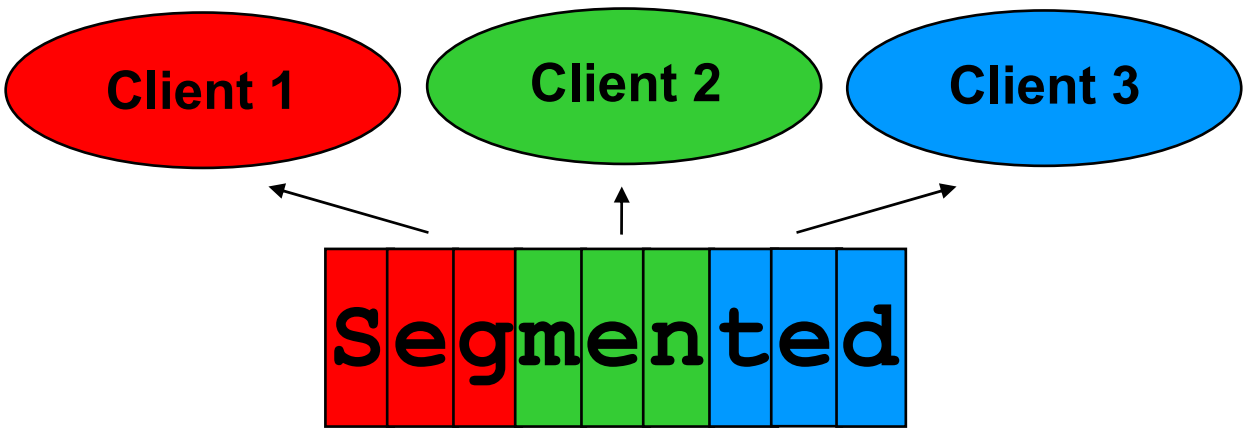
PART II:

- Performance
- File System Evaluation and Comparison
- Integrity
- GPFS on Linux
- Sysadm View/Tools/Tuning

Performance
(AIX, GPFS 1.4)

Test Description:

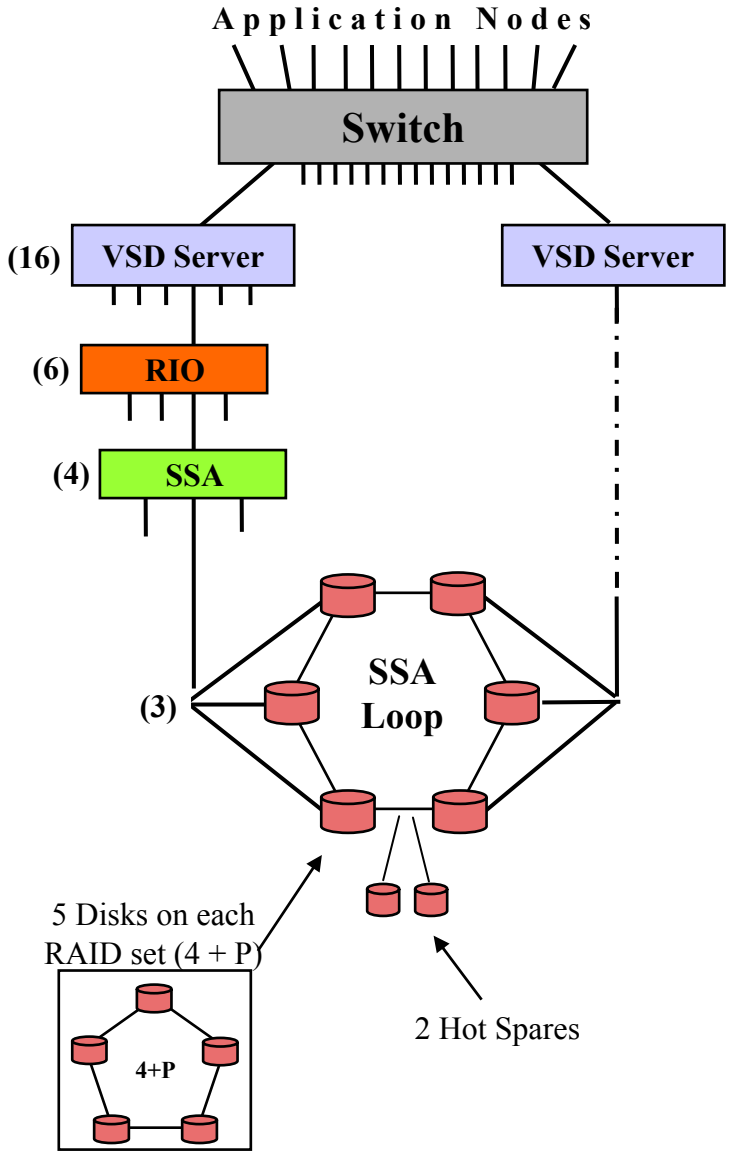
- Segmented data pattern
- Transfer size = GPFS block size = 512KB
- 1 client process per node
- vary nodes
- create, write, and read a single common file (size = 512MB*n)



Hardware

(on white – Fall, 2001)

- The Colony switch adapters are double/single.
- System software is Mohonk2 (PSSP 3.3) with GPFS 1.4.
- The tests discussed here were performed on white using up to 272 compute nodes and the GPFS file systems using 16 dedicated I/O nodes (servers).
- The tests performed utilize the POSIX interface to GPFS.



Tests

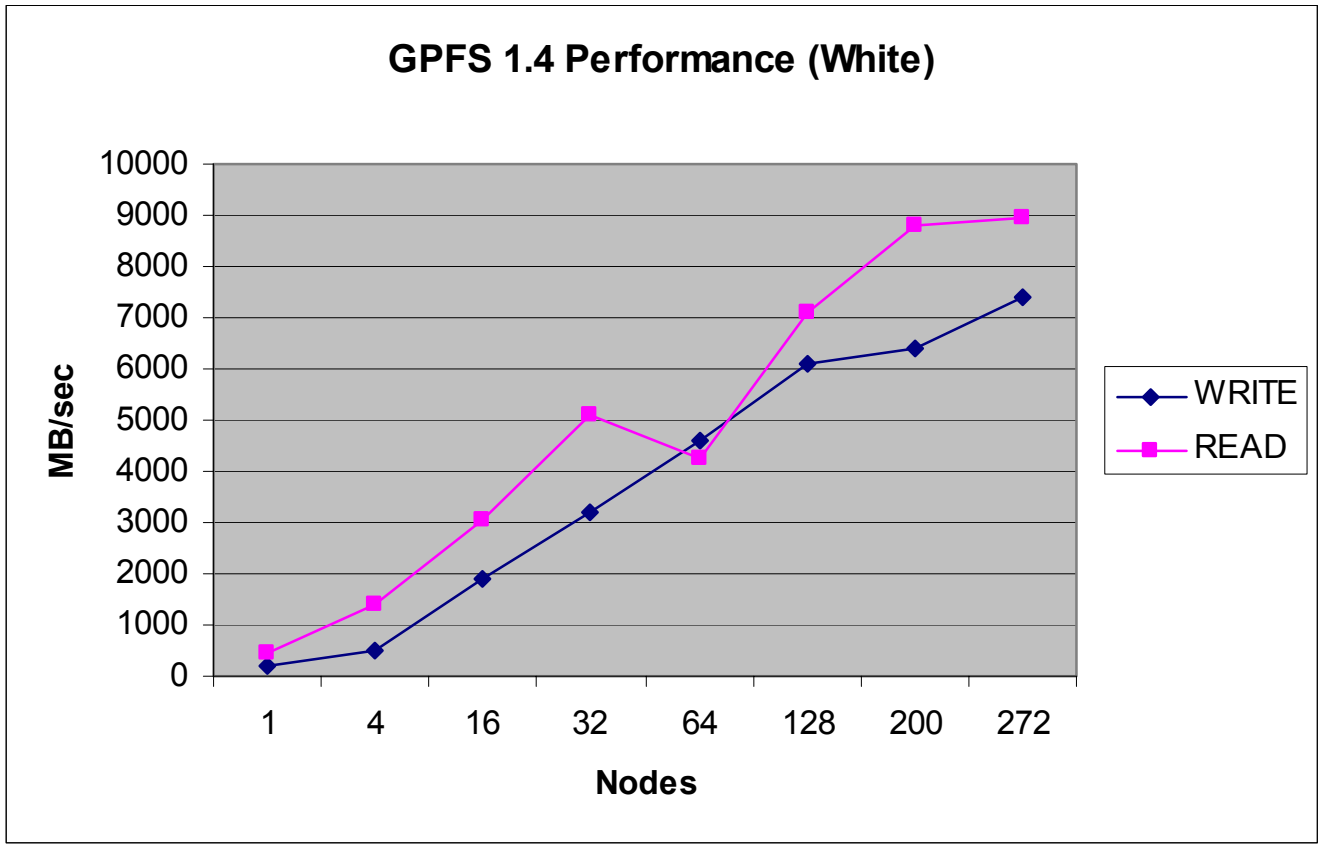
Each iteration of the test should erase an existing file (if any), create the file, write to the file, and close the file using POSIX calls. The time to erase, create, and close are not part of the timing for performance.

Each iteration of each test will have all clients open a single file on the GPFS. Each client will then write to this file in 512KB transfers until each has completed its BlockSize. The access will be in a segmented pattern.

- BlockSize – amount of data being written by an individual client. In these tests, this will be a contiguous segment of data.
- SubBlockSize – amount of data in the buffer for a single write call. It should match the stripe size of a RAID set.
- File size – product of the BlockSize and the number of clients.

- BlockSize = 512MB
- SubBlockSize = 512KB
- Iterations = 3
- TasksPerNode = 1
- Nodes = [1, 4, 16, 32, 64, 128, 200, 272]

Results


GPFS 1.4 Performance (White)
MB/sec
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
1
4
16
32
64
128
200
272
Nodes
WRITE
READ

Evaluation Chart

GPFS Design Goals

main advantage	parallel, high-performance
main disadvantage	IBM hardware only
special features	scalable

GPFS Synopsis

transparency	yes
user mobility	?
file mobility	?
managability	yes
limitations	?
scalability	yes
performance	yes

GPFS Architecture

name space	global access
versioning	?
semantics of sharing & synchronization	concurrent access
operations	POSIX
atomic units	byte

GPFS Implementation

client server	yes
networking	?
communication protocol	IP/KLAPI
locking	yes
caching and aggregation	yes
striping	yes
pathname traversal	POSIX
remote access method	?
availability	global
fault tolerance	yes
layered over other file system	no
allocation method (block vs. extent)	block

GPFS Managability

reconfiguration	yes
extensibility	yes
tuning	yes
problem logging	yes
installation	yes
debugging support	?

GPFS Limitations

maximum file system size	4 petabytes
maximum file size	2^{63}-1 bytes (~8 exabytes)
maximum number of logical volumes	32 file systems
maximum number of directories	?
maximum number of files	256 million

GPFS Linux Specific Issues

kernel patch	kernel extension

GPFS Performance
maximum asymptomatic data rate

writes on one I./O server	600 MB/sec
reads on one I./O server	800 MB/sec
writes on one client	180 MB/sec
reads on one client	450 MB/sec
block size at “elbow”	512KB
file size at “elbow”	Scalable

File System Comparison

	NFS	DFS	GFS	PVFS	GPFS
Scalability	N	Y	Y	Y	Y
Parallelism	N	N	N	Y	Y
Cross-platform	Y	Y	N	Y	N
Security	N	Y	N	N	Y
Failure recovery	N	Y	?	N	Y
Byte-range locking	Y	Y	Y	Y	Y

GPFS Integrity

Interface

- POSIX

Security

- uses standard user/group security for file access
- uses sysctl commands, secured through Kerberos and ACL files

Consistency

- managed by Token Server Manager
- note that with locking, the algorithm is to give 0-infinity for first task; then, if task 2 needs access from 1024-2047, give task 1 0-1023, and task 2 from 1024-infinity

Failure/Recovery

- each node must be on the switch and have mmfs running on it; if it crashes, the node is fenced off
- retry protocol in the VSD transport layer for each failure (dropped packet), waits 2x as long before retrying
- logs (1 or 2 copies maintained for each node) contains record of allocation and modification of metadata; used to assist with data consistency and recovery when GPFS node fails
- Recoverable VSD (RVSD) used in conjunction with the twin-tailed or loop cabling of SSAs, allowing for a fail-over. Without this, disks would not be available were a node to fail.
- replication – how many copies of a file to maintain (default=1, max=2)
- failure group – a collection of disks that share common access paths or adaptor connection and could all become unavailable through a single hardware failure.

Recovery

- High Availability Group Services (HAGS) – a component of RSCT (Reliable Scalable Cluster Technology) – notifies GPFS of failure of another GPFS daemon or other component failure

- GPFS recovers from node failure using notifications from HAGS. If necessary, new File System or Configuration Managers are selected and started.

Recovery

GPFS provides protection against these failures:

Node failure

- When an inoperative node is detected, GPFS fences it off. Recovery involves rebuilding metadata structures (which may have been modified during failure). If not enough nodes in quorum (quorum = nodeset / 2 + 1), then all GPFS nodes are restarted automatically.

Disk failure

- In the event of a disk failure, GPFS discontinues use of the disk and waits for an available state. Setting the replication > 1 can offer recoverability against a RAID failure.
- There are Metadata Replicas and Data Replicas settings. The allowable values are 1 and 2 (single- or double-copy), with default=1 for both. (Thus, with the loss of one RAID, the entire file system is lost.)

Connectivity failure

- This is controlled by creating environment-specific failure groups.

GPFS on Linux

- General move toward Beowulf Cluster computing (with Linux leading the way).

- But, Linux has had a hard time keeping up in the arena of parallel file systems. Lots of flops, but no where to write the data quickly.

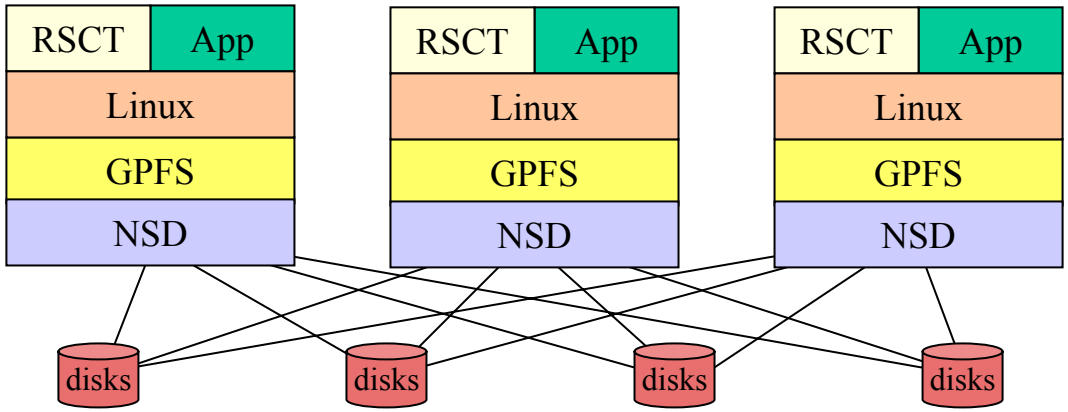
- IBM has ported GPFS to Linux to “ensure that GPFS will become the de-facto file system standard for Linux Clusters.”

GPFS on Linux Components

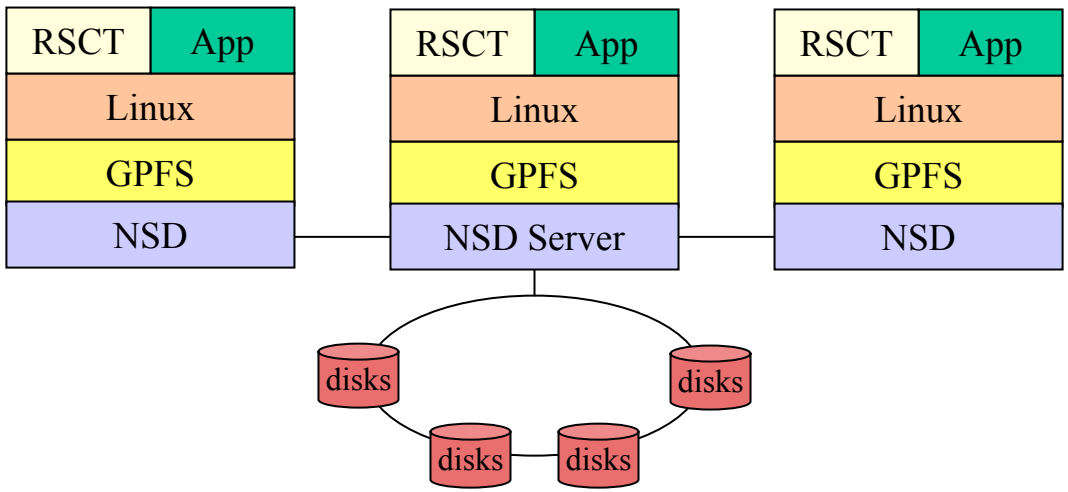
- Linux Kernel (Red Hat 7.1)
- RSCT (Reliable Scalable Cluster Technology) – subsystem of GPFS providing
  - HATS (High Availability Topology Services) – network services
  - HAGS (High Availability Group Services) – distributed coordination and synchronization
  - EM (Event Manager) – system resources
- GPFS (General Parallel File System)
- NSD (Network Shared Disk)

Two Designs

Either GPFS Cluster with disks directly attached:



or GPFS Cluster with NSD server:



Administrative Issues

- Installation
  - Not a turnkey process. Description of what is involved in initial installation/configuration of VSD servers and GPFS.
  - Scripts for automating installation.
  - GPFS command to define VSDs after setting up servers.
- Configuration/Tuning
  - Adding/Removing disks from file system
    - Rebalancing.
  - Additional Tuning

More Administrative Issues

- Administration
  - How much is automatic and how much must be done by hand (i.e., recovery, etc.)?
    - Since administration is distributed, a single administrative command on one node migrates to all other nodes (for example, a single ‘mount’ makes it possible to gain access to the entire file system no matter how many VSD servers.)
  - Addressing errors/problems
    - Recoverable file system error (server crash, etc.)
    - Unrecoverable file system error (data or metadata is lost)
  - mmfsadm – views mmfsd threads
  - mmfsck
- Other system administration issues and general problems
  - TCP communication timeouts
  - Disk Management – suspending writes to disks, migrating data off failing disks, etc.

Configuration/Tuning

- We have done alot of work in the tuning area in the past. The major thing was to make sure IO aligend with the disk block size. Thus the 256k base (we use 2x on white or 512k). This set by some of the lower level components namely LVM and SSA. We picked 4+p arrays because at 64k stripe width to each disk in the array 4*64=256k, SSA allows 7+p but that comes out to 488k and does not work well with GPFS. (Since an ssa drawer is max 16 drives doing something like 8+p does not make sense because you cannot create 2 8+p arrays in one drawer).
- Related to this is the max coalesce tuning which is an AIX LVM thing. It says how much data to group together into one IO op. This again is set to 256K the default is 128k.

Configuration/Tuning

- The other thing that we don't really mess with now is tuning the number of threads in a client. We did do this on the blue/sky machine before klapi. By limiting the number of threads available to do IO on the batch nodes we were able to better controll the break point at of client/server. Doing this also limits the single node performance, however these settings are per node. So, the login nodes were tuned up allowing for good single node performance.

Additional Information

– Bill Loewe <wel@llnl.gov>
– Mark Grondona <mgrondona@llnl.gov>

Special thanks to Dave Fox <foxd@llnl.gov>.